**Ensemble Methods**

We will learn how to take a bunch of models and join them together to get a better model. We will use methods called Bagging and Boosting.

**Bagging**

If we have 5 friends (weak learners) that all take a test for us. We see what answers they all answer more and take those as the answers (strong learner).

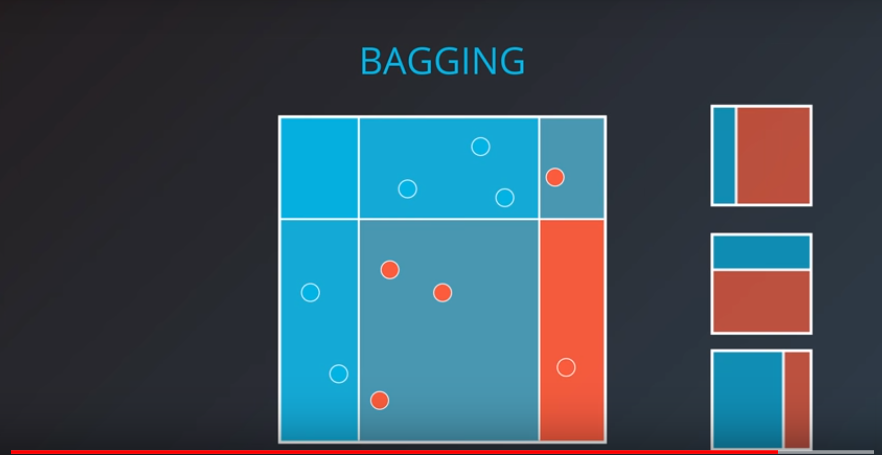
**Boosting**

With boosting we do something similar but we would denote certain testers to be better for certain answers (guys better for sports) and give them preference for those questions with sports.

These weak models don’t need to be that good to begin with (slightly better then randomly guessing is enough) but combined they can be very powerful.

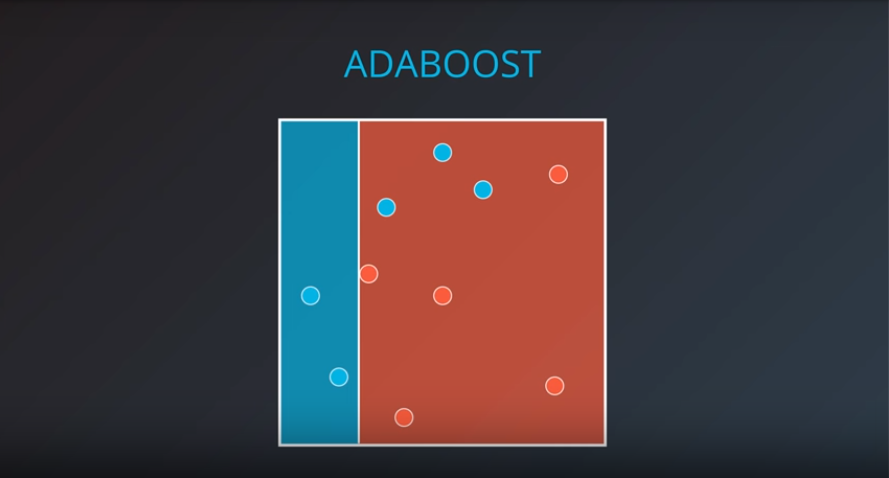
**Bagging extended**

Let’s say our weak learners are all decision trees of one node – split the data by one line. We can take random subsets of the data to create three weak learners and then take a vote on which sections should be red and should be blue:

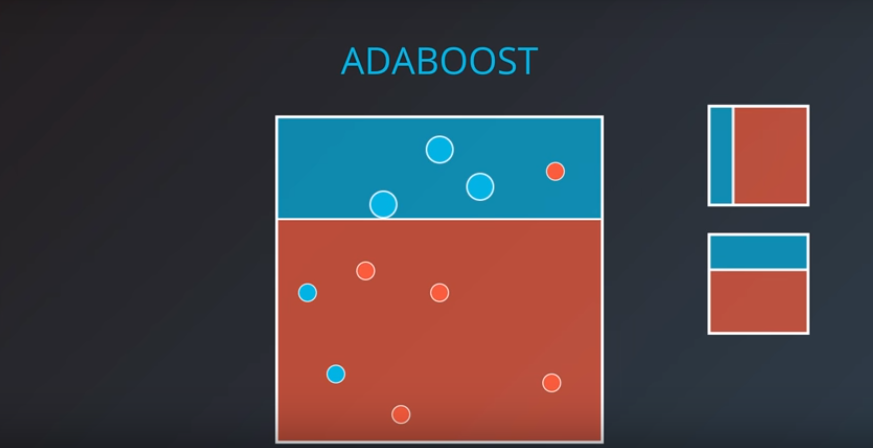


**Adaboost**

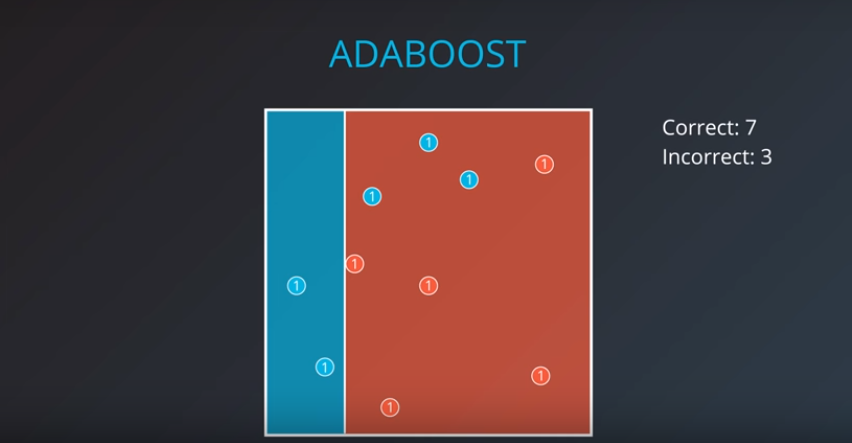
One of the more popular ways of doing boosting is Adaboosting. Adaboosting works as follows. Take a simple model and try to make as few mistakes as possible:



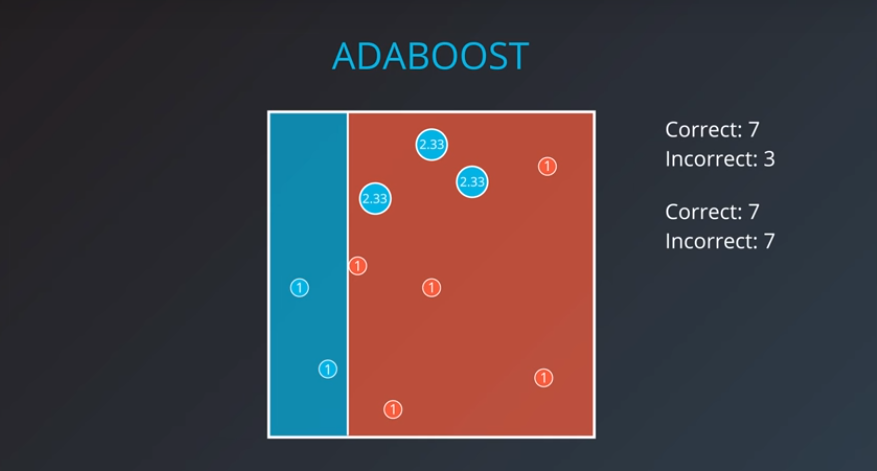
Here we have three errors, we’ll take these misclassified points and consider them to be weighted so we will want to definitely classify them correctly in our next weak model:



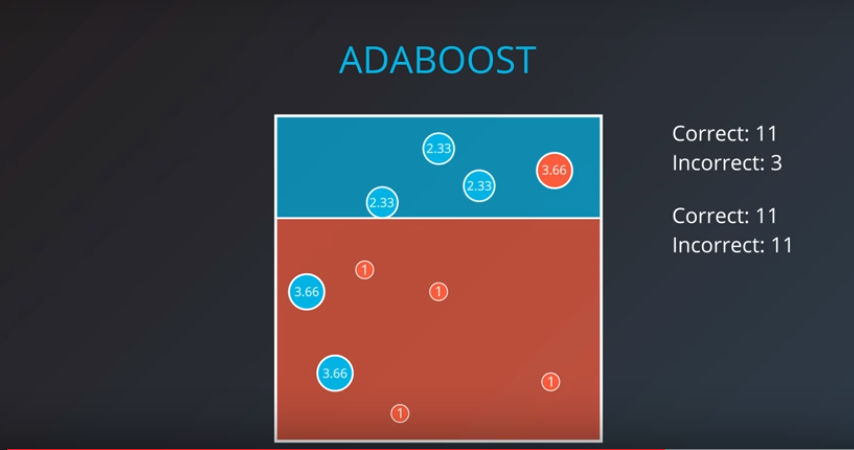
If we do this using numbers we can do a sum of the weights of all the incorrectly classified points. For example….



We’ll give the incorrectly classified points a weight which will mean incorrect and correct points will hold the same:

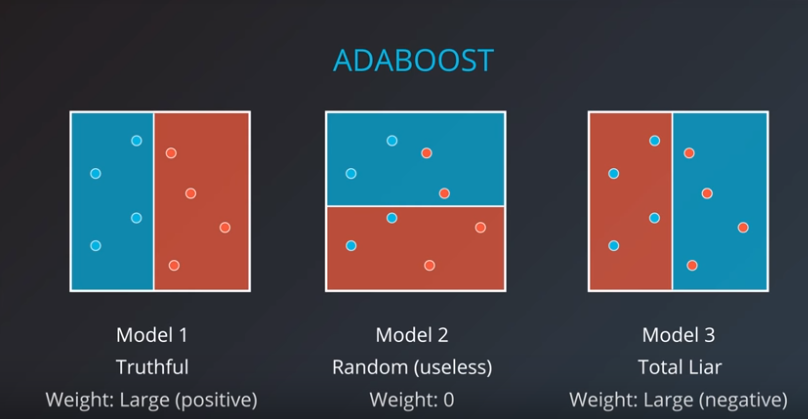


Second model we will do the same:

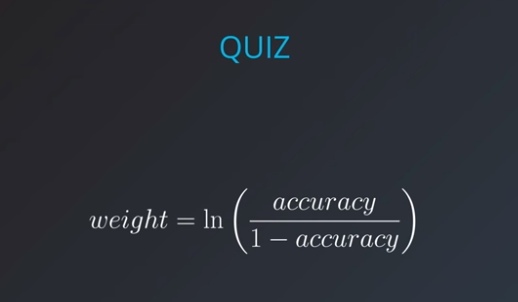


Third model we do the same and so on and so forth…

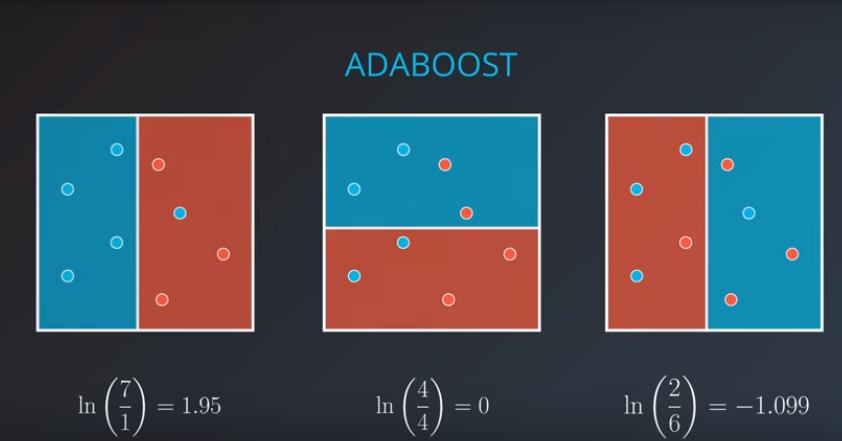
When it comes to weighting our weak models it’s best to weight them better if they are more definite (closer to being always right or always wrong).



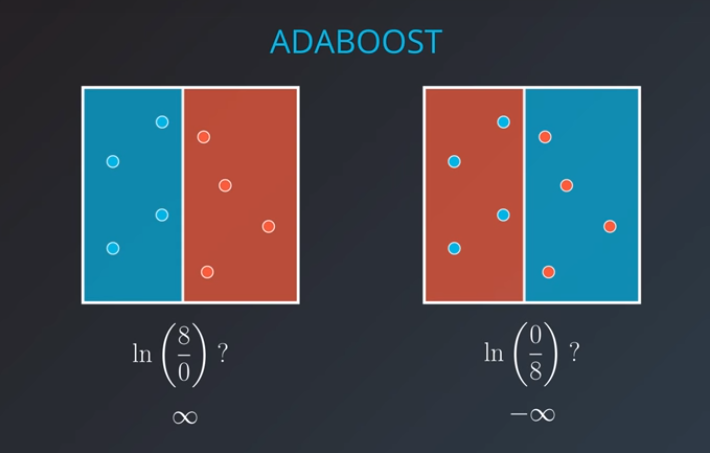
A good formula for weight is:



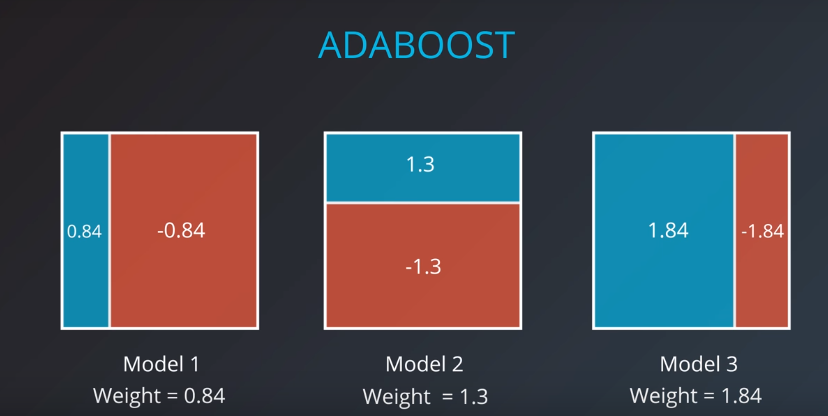
The weights for the below models would be:



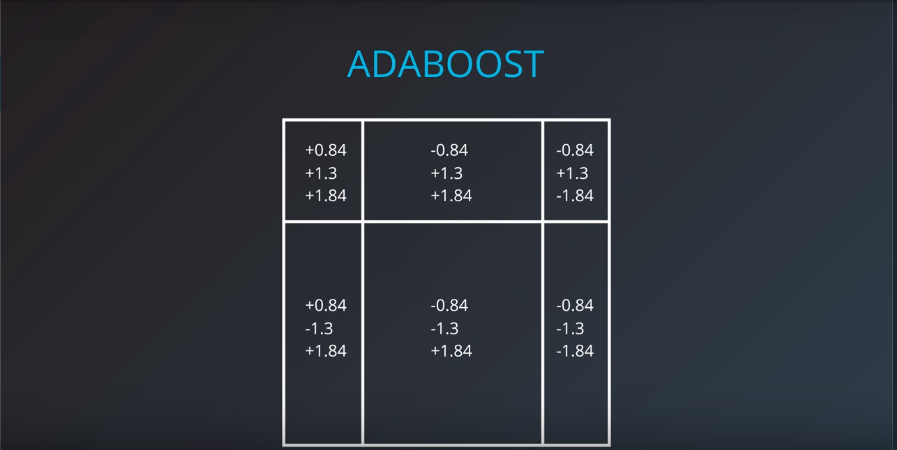
An example where we have a perfect weak learner:



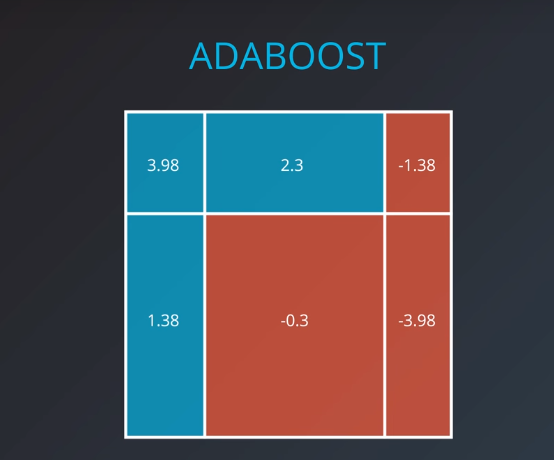
From our original example we would get the weights:



Combining these we get…



And calulating these sums we get:



**Adaboost in sklearn**

Building an AdaBoost model in sklearn is no different than building any other model. You can use scikit-learn's [AdaBoost Classifier](http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html) class. This class provides the functions to define and fit the model to your data.

>>> **from** sklearn.ensemble **import** AdaBoostClassifier

>>> model = AdaBoostClassifier()

>>> model.fit(x\_train, y\_train)

>>> model.predict(x\_test)

In the example above, the model variable is a decision tree model that has been fitted to the data x\_values and y\_values. The functions fit and predict work exactly as before.

**Hyperparameters**

When we define the model, we can specify the hyperparameters. In practice, the most common ones are

* base\_estimator: The model utilized for the weak learners (**Warning:** Don't forget to import the model that you decide to use for the weak learner).
* n\_estimators: The maximum number of weak learners used.

For example, here we define a model which uses decision trees of max\_depth 2 as the weak learners, and it allows a maximum of 4 of them.

>>> **from** sklearn.tree **import** DecisionTreeClassifier

>>> model = AdaBoostClassifier(base\_estimator = DecisionTreeClassifier(max\_depth=2), n\_estimators = 4)